PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-293057
(43)Date of publication of application : 11.11.1997

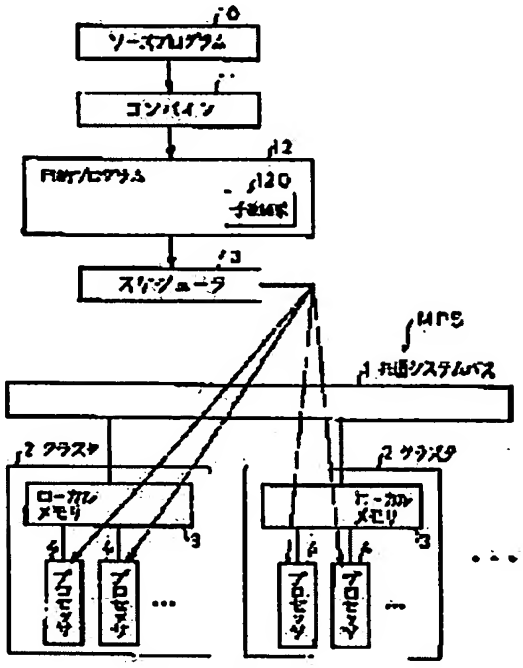
(51)Int.Cl. G06F 15/16

(21)Application number : 08-131124
(22)Date of filing : 26.04.1996

(71)Applicant : NEC CORP
(72)Inventor : FUNDOU KATSUAKI

(54) TASK ALLOCATION METHOD IN HIERARCHICAL STRUCTURE TYPE MULTIPROCESSOR SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To perform efficient parallel processing by the hierarchical structure type multiprocessor system.
SOLUTION: When a source program 10 is translated, a compiler 11 predicts the execution times of respective tasks which can be processed in parallel and overhead times at the time of the execution of the tasks by clusters different from a main cluster and incorporates them as prediction results 120 in an object program 12. A scheduler 13 when allocating the tasks to processors judges whether the tasks are completed earlier by expecting free processors to be obtained in the main cluster 2 or assigning the tasks to free processors in other clusters unless free processors to which all the tasks are assigned are present in the main cluster 2, and then allocates the tasks to the cluster which will complete the tasks.



LEGAL STATUS

[Date of request for examination] 26.04.1996
[Date of sending the examiner's decision of rejection] 23.03.1999
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－293057

(43)公開日　平成９年(1997)11月11日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/16	４３０		Ｇ０６Ｆ　15/16	４３０Ｂ

審査請求　有　　請求項の数３　ＦＤ　（全８頁）

(21)出願番号　特願平8－131124

(22)出願日　平成８年(1996)４月26日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　分銅　勝昭
東京都港区芝五丁目７番１号　日本電気株式会社内

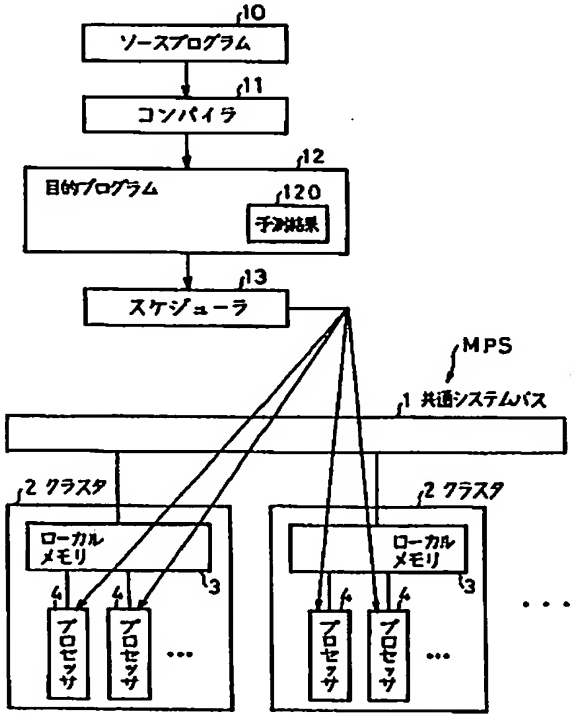
(74)代理人　弁理士　境　廣巳

(54)【発明の名称】　階層構造型マルチプロセッサシステムにおけるタスク割り当て方法

(57)【要約】

【課題】　階層構造型マルチプロセッサシステムにおいて効率の良い並列処理を実現する。

【解決手段】　ソースプログラム10の翻訳時、コンパイラ11は、並列処理可能な各タスクの実行時間と、タスクを主クラスタと異なるクラスタで実行した場合のオーバヘッド時間とを予測して目的プログラム12に予測結果120として組み込む。スケジューラ13は、タスクをプロセッサに割り当てる際に、並列処理可能な複数のタスクの全てを割り当てるだけの空きプロセッサが主クラスタ２に存在しない場合、予測結果120を利用して、主クラスタ２に空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを判断し、処理が早くなる側のクラスタにタスクを割り当てる。



1

【特許請求の範囲】

【請求項1】　メモリと該メモリを共有する複数のプロセッサとから構成されるクラスタを複数備え、これら複数のクラスタが共通システムバスを通じて相互にデータ転送可能に接続された階層構造型マルチプロセッサシステムにおけるタスク割り当て方法において、

ソースプログラムを翻訳して目的プログラムを生成するコンパイラにおいて、並列化の指示に従って並列処理可能な複数のタスクを生成した際に、各タスクの実行時間を予測すると共にタスクを前記目的プログラムを主として実行する主クラスタと異なるクラスタで実行した場合のオーバヘッド時間を予測して、これらの予測結果を目的プログラムに付加し、

目的プログラムのタスクをプロセッサに割り当てるスケジューラにおいて、並列処理可能なタスクは基本的に前記主クラスタのプロセッサ群に割り当てるようにするが、全タスクを割り当てるだけの空きプロセッサが不足している場合には、主クラスタに空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを、前記予測結果を利用して判断し、処理が早くなる側のクラスタにタスクを割り当てることを特徴とする階層構造型マルチプロセッサシステムにおけるタスク割り当て方法。

【請求項2】　タスクを前記主クラスタと異なるクラスタで実行した場合のオーバヘッド時間を、当該タスクを他のクラスタのプロセッサで実行させるためにそのクラスタのローカルメモリへ前記主クラスタのローカルメモリから転送しておく必要のあるデータの転送時間と、当該タスクの処理によって書き換えられた前記他のクラスタのローカルメモリのデータを前記主クラスタのローカルメモリに転送する時間とを、少なくとも考慮して予測することを特徴とする請求項1記載の階層構造型マルチプロセッサシステムにおけるタスク割り当て方法。

【請求項3】　並列処理可能な複数のタスクのうち次に割り当てようとするタスクのオーバヘッド時間と、並列処理可能な複数のタスクのうち既に主クラスタに割り当てられているタスクの実行時間および割り当て時刻から判明する空きプロセッサが生じるまでの待ち時間とを比較して、主クラスタに空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを判断することを特徴とする請求項1または2記載の階層構造型マルチプロセッサシステムにおけるタスク割り当て方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は階層構造型マルチプロセッサシステムで並列処理を行う場合のタスク割り当て方法に関する。

【０００２】

【従来の技術】マルチプロセッサシステムの一種に、図

2

6に示すように、複数のプロセッサ4とそれらで共有されるローカルメモリ3とから構成されるクラスタ2を複数備え、これら複数のクラスタ2を共通システムバス1によって相互にデータ転送可能に接続した階層構造型マルチプロセッサシステムがある。この階層構造型マルチプロセッサシステムの特徴は、各クラスタ2毎にローカルメモリ3を有する為、全てのプロセッサ4で1つのメモリを共有する通常の密結合マルチプロセッサシステムに比べてメモリアクセス競合の頻度が低下することである。但し、他のクラスタのローカルメモリ上のデータを必要とする場合には、そのデータを自クラスタのローカルメモリに転送する必要があり、時間がかかる。

【０００３】このような特徴を有するため、或るユーザプログラムを階層構造型マルチプロセッサシステムで実行する場合、複数のクラスタの中から1つのクラスタを主クラスタとして選択し、ユーザプログラムの開始から終了まで主クラスタ内のプロセッサに、ユーザプログラムを構成する各タスクを割り当てる方法が一般に採用されている。

【０００４】

【発明が解決しようとする課題】ところで、ユーザプログラムの或る処理を複数に分割してその各々を別々のタスクとしてマルチプロセッサシステムで並列処理する場合の、その分割方法として次の2通りがある。その1つは、例えば特開平1－152571号公報に示されるように、使用可能なプロセッサ台数を前提とし、ユーザプログラムの処理をその使用可能プロセッサ台数で均等に分割し、その各々を1つのタスクとする方法であり、他の方法は、ソースプログラム中でユーザが予め並列化数を指示しておき、その並列化指示に従ってユーザプログラムの処理を分割し、その各々を1つのタスクとする方法である。

【０００５】前者のような分割方法の場合、従来のタスク割り当て方法でも問題は生じない。同時に使用可能なプログラム台数を前提に並列処理するタスク数が決定されるため、全てのタスクが並列処理されるからである。しかしながら、後者のような分割方法の場合、タスク数分の空きプロセッサが常に存在するとは限らないため、全てのタスクが並列処理される保証はない。この場合、従来のタスク割り当て方法では、常に1つの主クラスタ内のプロセッサにタスクを割り当てるようにしている為、空きプロセッサ数が足りないと、空きプロセッサができるまで残りのタスクが必ず待たされることになる。

【０００６】なお、主クラスタ中に空きプロセッサが無い場合、他のクラスタに空きプロセッサがあれば直ちにそのプロセッサにタスクを割り当てるようにすることが考えられる。しかしながら、前述したように他のクラスタのローカルメモリ上のデータを必要とする場合にはそのデータを自クラスタのローカルメモリに転送する必要があって、それには或る程度の時間がかかるため、他の

3

クラスタのプロセッサに割り当てると、そのタスクの実行終了時刻や後続の処理の開始が却って遅くなる場合があり、問題である。

【０００７】そこで本発明の目的は、並列処理可能なタスクを基本的には1つのクラスタのプロセッサ群に割り当てるようにするが、全タスクを割り当てるだけの空きプロセッサが存在しない場合には、割り当て先を１つのクラスタに限定せず、処理が早まるならば別のクラスタに割り当てることで、ユーザプログラムの処理時間の短縮を図ることにある。

【０００８】

【課題を解決するための手段】本発明は、メモリと該メモリを共有する複数のプロセッサとから構成されるクラスタを複数備え、これら複数のクラスタが共通システムバスを通じて相互にデータ転送可能に接続された階層構造型マルチプロセッサシステムにおけるタスク割り当て方法において、ソースプログラムを翻訳して目的プログラムを生成するコンパイラにおいて、並列化の指示に従って並列処理可能な複数のタスクを生成した際に、各タスクの実行時間を予測すると共にタスクを前記目的プログラムを主として実行する主クラスタと異なるクラスタで実行した場合のオーバヘッド時間を予測して、これらの予測結果を目的プログラムに付加し、目的プログラムのタスクをプロセッサに割り当てるスケジューラにおいて、並列処理可能なタスクは基本的に前記主クラスタのプロセッサ群に割り当てるようにするが、全タスクを割り当てるだけの空きプロセッサが不足している場合には、主クラスタに空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを、前記予測結果を利用して判断し、処理が早くなる側のクラスタにタスクを割り当てることを特徴とする。

【０００９】また、タスクを前記主クラスタと異なるクラスタで実行した場合のオーバヘッド時間を、当該タスクを他のクラスタのプロセッサで実行させるためにそのクラスタのローカルメモリへ前記主クラスタのローカルメモリから転送しておく必要のあるデータの転送時間と、当該タスクの処理によって書き換えられた前記他のクラスタのローカルメモリのデータを前記主クラスタのローカルメモリに転送する時間とを少なくとも考慮して予測することを特徴とする。

【００１０】更に、並列処理可能な複数のタスクのうち次に割り当てようとするタスクのオーバヘッド時間と、並列処理可能な複数のタスクのうち既に主クラスタに割り当てられているタスクの実行時間および割り当て時刻から判明する空きプロセッサが生じるまでの待ち時間とを比較して、主クラスタに空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを判断することを特徴とする。

4

【００１１】

【発明の実施の形態】次に本発明の実施の形態の例について図面を参照して詳細に説明する。

【００１２】図1は本発明の一実施例のシステム構成図である。同図において、MPSは図6で説明した階層構造型マルチプロセッサシステムであり、各々が複数のプロセッサ4とそれらで共有されるローカルメモリ3とから構成される複数のクラスタ2が共通システムバス1によって相互にデータ転送可能に接続されている。また、１０はユーザが作成したソースプログラムである。このソースプログラム１０は高級言語で処理が記述されており、特に並列処理可能な部分についてはユーザが予めその並列化数を指示した並列化指示を埋め込んである。更に、１１はソースプログラム１０を翻訳して階層構造型マルチプロセッサシステムMPS向けの目的プログラム１２を生成するコンパイラである。このコンパイラ１１で生成される目的プログラム１２中には、コンパイラ１１の後述する処理によって、並列処理可能なタスクに関する実行時間とオーバヘッド時間との予測結果１２０が埋め込まれている。また、１３は目的プログラム１２を階層構造型マルチプロセッサシステムMPSで実行すべく、目的プログラム１２を構成する各タスクをクラスタ2のプロセッサ4に割り当てるスケジューラである。このスケジューラ１３は、目的プログラム１２を実行するために少なくとも１つの空きプロセッサが存在する１つのクラスタ2を主クラスタとして選択し、基本的に主クラスタ内のプロセッサ群に目的プログラム１２のタスクを割り当てる。但し、並列処理可能な複数のタスクを割り当てる際には割り当て先を主クラスタに限定せず、処理が早まるなら別のクラスタに割り当てる。

【００１３】図2はコンパイラ１１の構成例とその処理例とを示している。コンパイラ１１は、入力となるソースプログラム１０を読み込み、構文解析を行って中間テキスト１１３を生成する解析部１１０と、中間テキスト１１３を読み込み、ユーザから指示された並列化指示に従って処理を並列化して、並列化中間テキスト１１４を生成する並列化部１１１と、並列化中間テキスト１１４から目的プログラム１２を生成する生成部１１２とから構成され、特に並列化部１１１には、実行時間・オーバヘッド時間予測手段１１１０が設けられている。

【００１４】この実行時間・オーバヘッド時間予測手段１１１０は、並列化部１１１において、ソースプログラム１０中の或る処理部分が並列化指示に従って複数に分割されてその各々が1つのタスクとされた場合に、図2のステップＳ１～Ｓ３の処理を行って、各タスクを主クラスタのプロセッサで実行したときの実行時間と、主クラスタとは別のクラスタのプロセッサで実行したときのオーバヘッド時間とを予測する手段である。この予測された各タスクの実行時間とオーバヘッド時間とを含む予測結果は生成部１１２を通じて目的プログラム１２に予

5

測結果１２０として埋め込まれる。

【００１５】実行時間・オーバヘッド時間予測手段１１１０は、本実施例の場合、各タスクそれぞれに対して作成される命令の実行時間の総和（ループの場合には繰り返し回数も考慮にいれる）を求め、これを各タスクの実行時間とする（Ｓ１）。

【００１６】また実行時間・オーバヘッド時間予測手段１１１０は、本実施例の場合、オーバヘッド時間の大部分がデータ転送時間で占められることに鑑み、各タスクを主クラスタとは別のクラスタのプロセッサに割り当てた場合に、主クラスタとそのクラスタとの間で転送が必要となるデータのサイズを算出し（Ｓ２）、この算出したデータサイズに単位サイズ当たりのデータ転送時間を乗じることによりデータ転送時間の総和を求め、これをそのタスクのオーバヘッド時間とする（Ｓ３）。

【００１７】或るタスクを主クラスタと別のクラスタのプロセッサに割り当てた場合、主クラスタとそのクラスタとの間で転送が必要となるデータとしては、例えば次のようなものがある。

【００１８】１つは、当該タスクの処理を行う前段階として、テキストと、処理を行うのに最小限必要な入力データがある。これらのデータは、目的プログラムのこれまでの処理を行ってきた主クラスタのローカルメモリに存在しているため、そのローカルメモリから当該タスクが実行されるクラスタのローカルメモリへ転送しておく必要がある。

【００１９】他の1つは、当該タスクによって書き換えられるであろうと予測されるデータである。このようなデータは当該タスクが実行されるクラスタのローカルメモリ上で書き換えられるため、目的プログラムの主クラスタでの後続処理のために、主クラスタのローカルメモリへ転送しておく必要がある。

【００２０】以上のような種類のデータはソースプログラム１０のコンパイル時点で判明する為、実行時間・オーバヘッド時間予測手段１１１０はそれらのデータのサイズの総和を求め、そして、データ転送時間は転送されるデータのサイズにほぼ比例するので、算出したサイズに単位サイズ当たりのデータ転送時間を乗ずることにより、当該タスクを他のクラスタに割り当てた際のオーバヘッド時間を求める。

【００２１】なお、クラスタ間で転送が必要となるデータとしては、上述したデータが最もサイズが大きくなるため、それだけでオーバヘッド時間を求めるようにしても良いが、例えば、処理の途中で他のタスクと同期処理が必要な場合、同期をとらなくてはならないデータを転送する必要があるため、この種のデータの転送時間も考慮するようにしても良い。但し、この種のデータの転送は1度だけでなく複数回行われる場合があり、また双方向に転送されるので、それを考慮してデータ転送時間を求めることが必要である。

6

【００２２】次に図1のスケジューラ１３の動作について説明する。スケジューラ１３は、目的プログラム１２の実行に際して、複数のクラスタ２のうちから少なくとも１つの空きプロセッサが存在する１つのクラスタを、目的プログラム１２を実行する主クラスタとして選択する。そして、基本的に主クラスタ内のプロセッサ群に目的プログラム１２のタスクを割り当てる。但し、並列処理可能な複数のタスクを割り当てる際に空きプロセッサが不足する場合は、主クラスタでプロセッサが空き状態になるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを目的プログラム１２中の予測結果１２０を利用して、処理が早くなる方にタスクを割り当てる。なお、並列処理可能な複数のタスクのうち最初に割り当てるタスクは必ず主クラスタのプロセッサに割り当てるため、若し、空きプロセッサが存在しなければ待ち状態となる。

【００２３】図3はスケジューラ１３が並列処理可能な複数のタスクを順次に割り当てる際に2番目以降のタスクについて行う処理の一例を示している。同図に示すように、スケジューラ１３は、並列処理可能な次のタスクを割り当てる際、先ず、主クラスタ中のプロセッサの使用状況を確認し、空きプロセッサが存在するか否かを調べる（Ｓ１１）。主クラスタ中に空きプロセッサが存在する場合には、そのプロセッサに当該タスクを割り当てる（Ｓ１６）。

【００２４】他方、主クラスタ中に空きプロセッサが存在しない場合、主クラスタに空きプロセッサが生じるを待ってその空きプロセッサに当該タスクを割り当てた方が処理が早いか、それとも他のクラスタの空きプロセッサに当該タスクを割り当てた方が処理が早いかを判定する。このために、先ず、主クラスタに空きプロセッサができるまでの待ち時間Xを予測する（Ｓ１２）。この待ち時間Xは、当該並列処理可能な複数のタスクであって主クラスタに既に割り当てられたタスクのうち、最も早く実行を終えるタスクの終了時刻までの時間とする。タスクの実行終了時刻は、その割り付け時刻に予測結果１２０中に記述されたそのタスクの実行時間を加算して求める。次に、今回割り当てようとするタスクのオーバヘッド時間（これも予測結果１２０中に記述されている）が、上記待ち時間Xより小さいか否かを判定する（Ｓ１３）。オーバヘッド時間が待ち時間Xより小さい場合、他のクラスタの空きプロセッサに当該タスクを割り当てた方が処理が早まる可能性が高いので、他のクラスタに空きプロセッサが存在するか否かを調べ（Ｓ１４）、存在すればその空きプロセッサに当該タスクを割り当てる（Ｓ１５）。なお、他のクラスタに空きプロセッサが存在しなかった場合にはステップＳ１１に戻って上述した処理を繰り返す。一方、オーバヘッド時間が待ち時間Xより小さくない場合は、主クラスタに空きプロセッサが生じるのを待っていた方が処理が早まる可能性が高いの

7

で、ステップＳ１１に戻って上述した処理を繰り返す。

【００２５】例えば図4（Ａ）に示すように処理A，処理B，処理Cから構成されるソースプログラム１０をコンパイラ１１が翻訳した結果、図4（Ｂ）に示すような処理Aを実行するタスクＴＡ，処理Bを並列実行する３つのタスクＴＢ０，ＴＢ１，ＴＢ２，処理Cを実行するタスクＴＣから構成される目的プログラム１２が生成され、主クラスタの或るプロセッサでタスクＴＡが実行され、その終了を契機に3 つのタスクＴＢ０，ＴＢ１，ＴＢ２を割り当てる場合、主クラスタに3 つの空きプロセッサがあれば、図5（Ａ）に示すようにタスクＴＢ０，ＴＢ１，ＴＢ２の全てが主クラスタ内のプロセッサに割り当てられる。しかし、例えば空きプロセッサが2 つしかないと、図5（Ｂ）に示すようにタスクＴＢ０，ＴＢ１を主クラスタの空きプロセッサに割り当てた時点で、空きプロセッサが無くなる。そこで、スケジューラ１３は、タスクＴＢ２の割り当てにかかる図3 の処理において、主クラスタに空きプロセッサが存在しないことを判定すると（Ｓ１１）、主クラスタに空きプロセッサができまるでの待ち時間Xを予測する（Ｓ１２）。現在の時刻をタスクＴＢ２の割り当て直後の時刻とすると、その待ち時間X は図5（Ｂ）に示す時間Xとなる。そこで、スケジューラ１３は、この待ち時間Xとタスクの時間（図5（Ｂ）のa +b の時間）との大小関係を調べる（Ｓ１３）。そして、オーバヘッド時間が待ち時間Xより小さい場合には、図5（Ｂ）に示すように、他のクラスタに空きプロセッサがあればそのプロセッサにタスクＴＢ２を割り当てる。こうすると、主クラスタに空きが生じるのを待ってタスクＴＢ２を割り当てる場合に比べて、図5（Ｂ）に示す時間Yだけ処理時間が短縮できることになる。なお、他のクラスタに空きプロセッサが存在しない為に、ステップＳ１１～Ｓ１４のループを繰り返していると、待ち時間X は徐々に短くなるのに対してオーバヘッド時間は固定なので、ついには待ち時間X がオーバヘッド時間以下になる。こうなると、タスクＴＢ２はもはや他クラスタへは割り当てられず、主クラスタに空きプロセッサが生じるのを待つことになる。

【００２６】以上本発明の実施例について説明したが、本発明は以上の実施例にのみ限定されず、その他各種の付加変更が可能である。例えば、上述した実施例では、並列可能な複数のタスクの各々についてオーバヘッド時間を予測したが、並列処理可能な複数のタスクの割り当て順序が定まっており、前述したように最初のタスクを必ず主クラスタに割り当てるようにしている場合にあっては、並列処理可能な複数のタスクのうちその先頭のタスクについてはオーバヘッド時間の予測を省略することができる。

8

【００２７】

【発明の効果】以上説明したように本発明によれば、ソースプログラムの翻訳時点で、並列処理可能な各タスクの実行時間と、タスクを主クラスタと異なるクラスタで実行した場合のオーバヘッド時間とを予測して目的プログラムに含めておき、スケジューラがタスクをプロセッサに割り当てる際に、並列処理可能な複数のタスクの全てを割り当てるだけの空きプロセッサが主クラスタに存在しない場合、目的プログラムに含められた予測結果を利用して、主クラスタに空きプロセッサが生じるのを待った方が早いか、それとも他のクラスタの空きプロセッサにタスクを割り当てた方が早いかを判断し、処理が早くなる側のクラスタにタスクを割り当てるようにしたので、より効率の良い並列処理が可能となる。

【図面の簡単な説明】

【図1】本発明の一実施例のシステム構成図である。

【図2】コンパイラの構成例と処理例とを示す図である。

【図3】スケジューラの処理例を示すフローチャートである。

【図4】ソースプログラムと目的プログラムの例を示す図である。

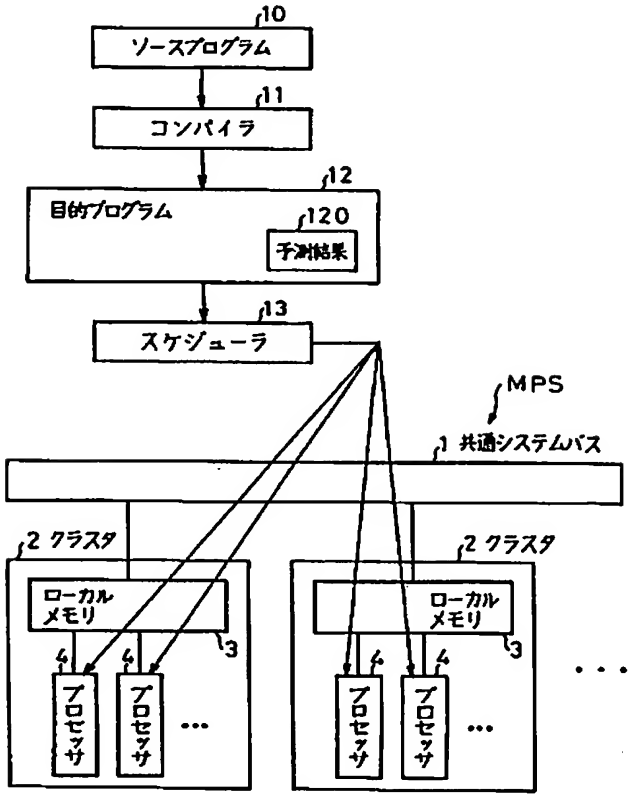
【図5】並列処理可能なタスクのクラスタへの割り当てと実行時間との関係を示す図である。

【図6】階層構造型マルチプロセッサシステムの一例を示すブロック図である。

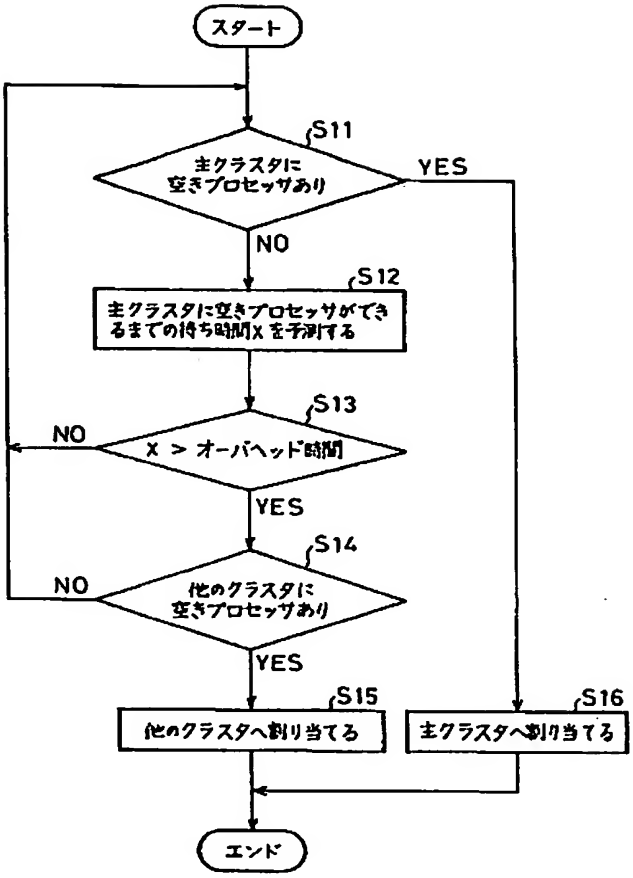
【符号の説明】

MPS…マルチプロセッサシステム
１…共通システムバス
２…クラスタ
３…ローカルメモリ
４…プロセッサ
１０…ソースプログラム
１１…コンパイラ
１２…目的プログラム
１２０…予測結果
１３…スケジューラ

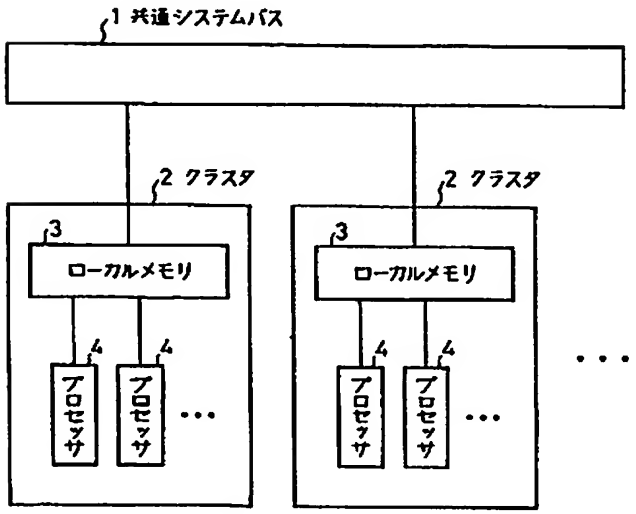
【 図1 】


10
ソースプログラム
11
コンパイラ
12
目的プログラム
120
予測結果
13
スケジューラ
MPS
1 共通システムバス
2 クラスタ
ローカルメモリ
3
4
プロセッサ
2 クラスタ
ローカルメモリ
3
4
プロセッサ

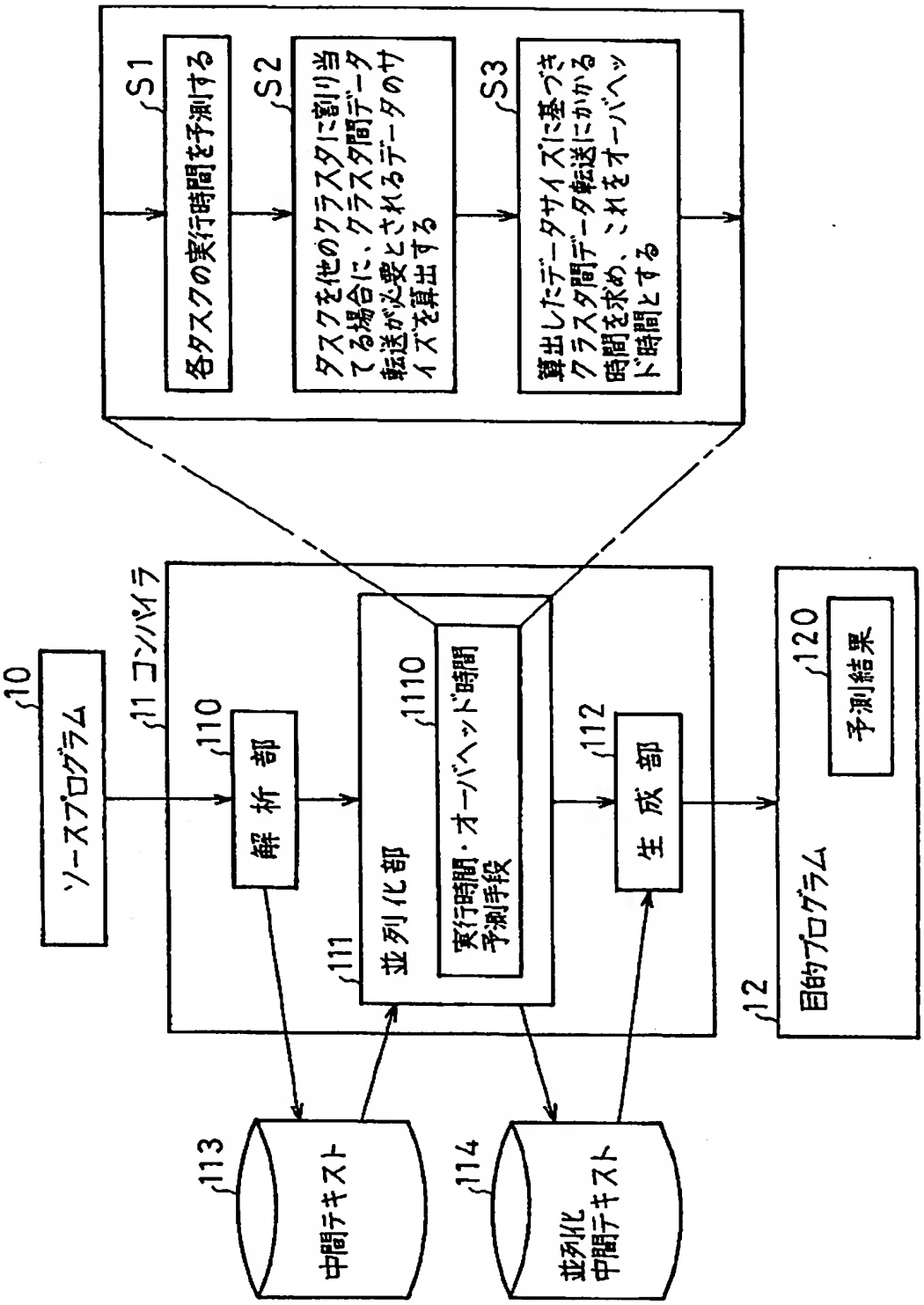
【 図3 】


スタート
S11
主クラスタに空きプロセッサあり
YES
NO
S12
主クラスタに空きプロセッサができるまでの待ち時間Xを予測する
S13
NO
X > オーバヘッド時間
YES
S14
NO
他のクラスタに空きプロセッサあり
YES
S15
他のクラスタへ割り当てる
S16
主クラスタへ割り当てる
エンド

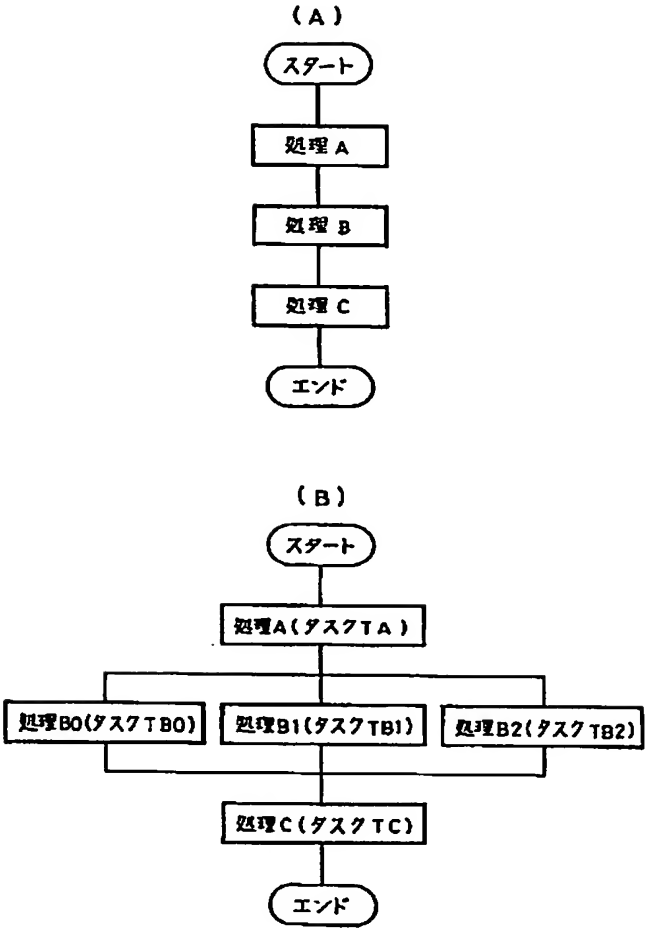
【 図6 】


1 共通システムバス
2 クラスタ
3
ローカルメモリ
4
プロセッサ
2 クラスタ
3
ローカルメモリ
4
プロセッサ

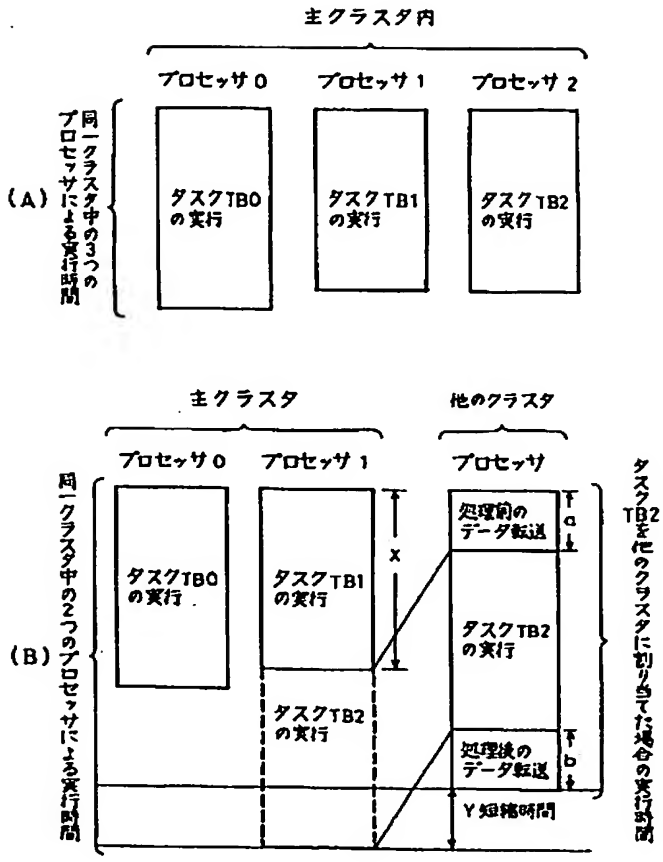
【 図2 】


S1
各タスクの実行時間を予測する
S2
タスクを他のクラスタに割り当てる場合に、クラスタ間データ転送が必要とされるデータのサイズを算出する
S3
算出したデータサイズに基づきクラスタ間データ転送にかかる時間を求め、これをオーバヘッド時間とする
10
ソースプログラム
11 コンパイラ
110
解析部
1110
実行時間・オーバヘッド時間予測手段
111
並列化部
112
生成部
120
予測結果
12
目的プログラム
113
中間テキスト
114
並列化中間テキスト

【 図4 】


(A)
スタート
処理 A
処理 B
処理 C
エンド
(B)
スタート
処理A(タスクTA)
処理B0(タスクTB0)
処理B1(タスクTB1)
処理B2(タスクTB2)
処理C(タスクTC)
エンド

【 図5 】


主クラスタ内
プロセッサ0
プロセッサ1
プロセッサ2
(A)
同一クラスタ中の3つのプロセッサによる実行時間
タスクTB0の実行
タスクTB1の実行
タスクTB2の実行
主クラスタ
他のクラスタ
プロセッサ0
プロセッサ1
プロセッサ
(B)
同一クラスタ中の2つのプロセッサによる実行時間
タスクTB0の実行
タスクTB1の実行
タスクTB2の実行
処理前のデータ転送
処理後のデータ転送
Ta
Tb
X
Y短縮時間
タスクTB2を他のクラスタに割り当てた場合の実行時間